02

CHAPTER

開発環境の準備
LLM と一緒に

プログラミング学習で最初の、そして最大の壁を乗り越える

本書のサンプル章 — 全文無料公開

AGENDA

この章で学ぶこと

01 **Python 3.13+, Git, Cursor のインストール**
OS ごとの注意点を押さえる

02 **Cursor のセットアップ**
AI 統合エディタを使いこなす

03 **ターミナル / コマンドラインの基礎**
最小限のコマンドだけで OK

04 **Git の基本操作**
「コードのバックアップ」と考える

05 **uv で仮想環境を構築**
高速・モダンな新定番

06 **つまずきポイント集と LLM 活用**
エラーを学習のチャンスに

INTRODUCTION

環境構築こそが、最大の壁

初心者がぶつかる「謎のエラー」

「PATH って何？」

「permission denied って何？」

「python コマンドが見つからない！？」

Google 検索を繰り返し、数時間後に疲れ果てて諦めてしまう。

本章のアプローチ

レシピ通りに料理を作るように

- ✓ **つまずきポイントを先に教える**
  ハマる前に避けられる
- ✓ **LLM に即聞ける質問例を提示**
  詰まったら数秒で抜け出す
- ✓ **詳細手順は公式 / LLM に任せる**
  最新情報・ OS 差分に強い
- ✓ **一つずつ確実に**
  焦らない、深呼吸して

TOOLS

必要なツール（基本すべて無料）

Cursor のみ、本格利用は有料プラン推奨

TOOL

Python 3.13+

プログラミング言語

Win / Mac

Free

TOOL

Git

バージョン管理

Win / Mac

Free

TOOL

Cursor

AI 統合エディタ

Win / Mac

Free / $20 /月

TOOL

Terminal

コマンド実行の窓口

標準搭載

Free

WINDOWS

Windows での準備

STEP 1

Python 3.13

python.org からダウンロード

✓ 「 **Add Python to PATH** 」に必ずチェック

STEP 2

Git

git-scm.com からダウンロード

デフォルト設定で **OK**
インストール後 **PowerShell** 再起動

STEP 3

Cursor

cursor.com からダウンロード

セットアップウィザードを完了

INSTALL CHECK (PowerShell)

```
PS> python --version    PS> git --version
```

macOS での準備（ Homebrew 経由 ）

STEP 1

Homebrew

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

M1/M2 Mac は完了後の 2 コマンドを忘れずに

STEP 2

Python 3.13

```
brew install python@3.13
```

確認 : python3 --version

STEP 3

Git (最新版)

```
brew install git
```

macOS には標準で入っているが最新版を推奨

STEP 4

Cursor

```
brew install --cask cursor
```

公式サイトから .dmg でも OK

PART 1

Cursor を使いこなす

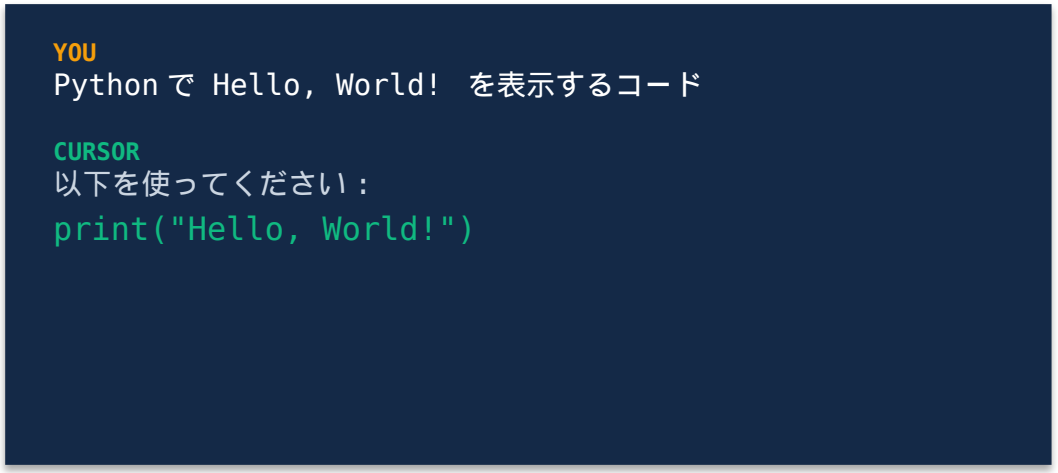
AI 統合エディタとの対話で、つまずきを乗り越える

AI との 2 つの対話チャネル

CHAT パネル

対話してコードを生成

`⌘ + L  /  Ctrl + L`



INLINE EDIT

選択コードを AI に編集させる

`⌘ + K  /  Ctrl + K`

- コードを選択して、自然言語で「こう変えて」
- エラー部分を選択して、修正案を提案させる
- コメントを書いて、それを実装させる
- リファクタリング・命名・型付けを依頼

CURSOR

コードベースのインデックス化と使い方のコツ

インデックス化とは

AI がコードを学ぶ仕組み

プロジェクトを開くと、 Cursor が自動的にコードを学習します。学習が進むほど、 AI の提案は賢くなります。

進行状況の確認

```
⌘+Shift+J  →  Indexing & Docs
```

目安: プロジェクトの規模により 1 〜 15 分

完了前は AI 機能が完全に動作しないことがあります

→ 100% を待ってから本格的に使い始める

TIPS

使い方の 4 つのコツ

1. **コードを選択 → ⌘ +K**
   AI に直接「こう変えて」と頼める
2. **Chat (⌘+L) で対話的に生成**
   要件を会話して詰めていく
3. **エラーは丸ごと貼り付け**
   メッセージそのままが最強の質問
4. **インデックス化を待つ**
   賢さの伸びしろが大きく変わる

PART 2

ターミナル & Git

最小限のコマンドだけ覚えれば十分

TERMINAL

本書で使う最小限のコマンド

pwd

現在地を表示

Print Working Directory

ls

ファイル一覧

List

cd

ディレクトリ移動

Change Directory

mkdir

フォルダ作成

Make Directory

terminal — bash

```
$ pwd
/Users/you

$ mkdir adventure-game
$ cd adventure-game
$ pwd
/Users/you/adventure-game

$ ls
#  空っぽ。ここからプロジェクトを始める

#  迷ったらホームに戻る
$ cd ~
```

わからない1コマンドはLLMにそのまま聞く = 最速の解決ルート

Git は「コードのバックアップシステム」

WHAT IT IS

コードの
バージョン管理システム

つまり、コードの変更履歴を記録して、いつでも過去の状態に戻せる仕組み。

例えるなら

「写真を撮って、過去のあの瞬間に戻れるアルバム」
壊しても、消しても、いつでもあの時に巻き戻せる。

Git でできる 5 つのこと

バックアップ	コードの状態を保存する
リストア	過去の状態に戻す
変更履歴	いつ何を変えたかが分かる
安全な実験	失敗しても戻せる
共有	他の人とコードを共有する

GIT WORKFLOW

4 つのコマンドで「保存 → 巻き戻し」

1

`git init`

リポジトリを初期化

アルバムを作る

2

`git add .`

ステージング

今回の写真を選ぶ

3

`git commit -m "…"`

バックアップを取る

シャッターを押す

4

`git log`

履歴を見る

アルバムをめくる

FIRST-TIME SETUP

Git の初期設定（最初の 1 回だけ）

```
$ git config --global user.name "Your Name"
$ git config --global user.email "you@example.com"
```

PART 3

仮想環境 with uv

高速・モダンな新定番でつまずきを根本回避

WHY uv

従来の venv + pip からの卒業

SPEED

10–100x

pip / venv より高速

統合 仮想環境とパッケージ管理を一元化

正確 依存関係の解決が正確

優しい 「有効化忘れ」を回避できる

従来の方法 vs uv

BEFORE — venv + pip

```
# OS 別に違うコマンド
$ source .venv/bin/activate      # macOS
$ .venv\Scripts\Activate.ps1    # Windows
$ python manage.py runserver
$ deactivate                     #  忘れがち
```

AFTER — uv run

```
$ uv run python manage.py runserver
```

OS 差なし・有効化不要・1 コマンド。初心者最大のつまずきが消える。

プロジェクト作成の鉄板パターン

01 INSTALL

uv をインストール

```
macOS: curl -LsSf https://astral.sh/uv/install.sh | sh
Win:  irm https://astral.sh/uv/install.ps1 | iex
```

02 INIT

プロジェクト初期化

```
$ uv init test-uv
$ cd test-uv
```

03 ADD

依存関係を追加

```
$ uv add django
# pyproject.toml に記録 + 仮想環境にインストール
```

04 RUN

コマンド実行

```
$ uv run python -m django --version
#  有効化不要！
```

覚えるのは「 init → add → run 」の 3 つだけ。

TROUBLESHOOTING

つまずきポイント TOP 6

01 PATH 設定の問題

```
command not found: python
```

→ Win 再インストール時に PATH を追加 / Mac は python3

02 権限エラー

```
Permission denied
```

→ sudo に頼らず仮想環境を使う

03 仮想環境が有効化されない

```
プロンプトに (.venv) が出ない
```

→ which python / $VIRTUAL_ENV を確認

04 Git リポジトリ未初期化

```
fatal: not a git repository
```

→ git init を忘れずに

05 Git ユーザー情報未設定

```
Please tell me who you are
```

→ git config --global user.name / .email

06 uv が動かない

```
command not found: uv
```

→ ターミナル再起動 / source $HOME/.cargo/env

READY?

環境構築おつかれさま。

ここまで来たあなたは、もう立派な開発者の仲間入り。

環境構築チェックリスト

- ✓ python --version が動く
- ✓ git --version が動く
- ✓ uv --version が動く
- ✓ Cursor が起動し AI 機能が使える
- ✓ ターミナルの基本コマンドが分かる
- ✓ エラー時の LLM 質問テンプレが手元にある

NEXT

第 3 章

Django プロジェクトを始める

プロジェクトを作って、初めての
「 Hello, World! 」を表示します。

エラーは誰にでも起こる。恐れず、一歩ずつ進みましょう。